XCELL

THE NEWSLETTER FOR XILINX PROGRAMMABLE LOGIC USERS

Issue 7 Second Quarter 1992

XCELL is back!

After a one-year intermission caused by lack of editing time (certainly not by lack of material), XCELL is back, and will again bring you technical information and timely availability updates on Xilinx devices and development software.

The XC4000 family now has three devices in full production, with one more available in sample quantity. Many new packaging and speed options for XC3000 and XC4000 devices are now available (see page 2).

Early this year, we acquired Plus Logic, and made it the EPLD division of Xilinx, dedicated to Programmable Logic Devices based on EPROM technology. The AND/OR structure and predictable performance of these devices are popular with designers accustomed to PAL* devices. The two presently available Xilinx devices offer interesting advantages in speed and density.

Xilinx development systems have made great strides to become more powerful and easier to use.

We have also improved our applications support and have published the first version of a Xilinx Applications Handbook, appropriately called XAPP.

Expect more devices, new technologies, new and better software, additional interfaces and platform support from Xilinx and third-party vendors. And expect to read about it in XCELL.

Welcome back!

Peter Alfke, Editor

Two New Software Tools: X-BLOX and Xilinx ABEL

Xilinx has recently introduced two new software tools that shorten design cycles, and increase productivity. X-BLOX™ and Xilinx ABEL software tools provide intelligent, high-level design entry capabilities that significantly reduce the need for gate-level design.

Using X-BLOX tools with existing design-entry tools, logic can be designed at the block-diagram level. Functional blocks, like adders and registers, can be specified in a schematic, and incorporated directly into the design without detailed logic design. The X-BLOX synthesis design tool, however, is much more than a macro library.

A rule-based expert system creates optimized Hard Macros for each of the X-BLOX modules in the schematic. Data paths of any width are automatically accommodated, and the XC4000 dedicated carry logic is used wherever appropriate. The result is an XNF file that can be placed and routed just like any other.

Xilinx ABEL also augments existing design entry tools, by permitting functional blocks to be defined using the industry standard ABEL** Hardware Description Language instead of gates. As with X-BLOX tools, the resulting XNF file can be placed and routed using the standard Xilinx tools.

Targeted primarily at automated state-machine design, Xilinx ABEL uses One-Hot encoding. This technique exploits the large number of flip-flops contained in LCA devices, while minimizing the impact of the CLB fan-in restriction. One-Hot encoding usually provides the highest performance state machines.

More detailed descriptions of X-BLOX and Xilinx ABEL software tools appear later in this issue.

* PAL is a trademark of AMD.
** ABEL is a trademark of Data I/O Corp.

Table of Contents

LCA Component Availability (July 1990)

		44 PIN	68 PIN		84 PIN		100 PIN			120 PIN	132 PIN		156 PIN	160 PIN	164 PIN		175 PIN		191 PIN	196 PIN	208 PIN
		PLASTIC PLCC	PLASTIC PLCC	CERAMIC PGA	PLASTIC PLCC	CERAMIC PGA	PLASTIC PQFP	CERAMIC CQFP	PLASTIC TQFP	CERAMIC PGA	PLASTIC PGA	CERAMIC CQFP	CERAMIC PGA	PLASTIC PQFP	CERAMIC CQFP	TOP BRAZED CERAMIC CQFP	PLASTIC PGA	CERAMIC PGA	CERAMIC PGA	CERAMIC CQFP	PLASTIC PQFP
		PC44	PC68	PG68	PC84	PG84	PQ100	CQ100	TQ100	PG120	PP132	PG132	PG156	PQ160	CQ164	CQ164	PP175	PG175	PG191	CQ196	PQ208
XC2064	-50	CI	CI	CIM																	
	-70	CI	CI	CIM																	
	-100	C	C	C																	
XC2018	-33					MB															
	-50	CI	CI		CI	CIMB															
	-70	CI	CI		CI	CIMB															
	-100	C	C		C	C			C												
XC3020	-70		CI		CI	CIMB	CI	CMB													
	-100		CI		CI	CI	C	C													
	-125		C		C	C	C														
	-150				C																
XC3030	-70	CI	CI		CI	CIM	C														
	-100	C	CI		CI	CI	C		C												
	-125	C	C		C	C	C														
	-150				C																
XC3042	-70				CI	CIMB	CI	CMB			CI	CI									
	-100				CI	CI	CI	C	C		CI	CI									
	-125				CI	CI	CI	C			CI	CI									
	-150				C																
XC3064	-70				C						CI	CIM		C							
	-100				CI						CI	CI		C							
	-125				C						C	C		C							
	-150				C																
XC3090	-70				CI									CI	CMB	MB	CI	CIMB	CI	CI	CI
	-100				CI									CI	C		CI	CI	CI	CI	CI
	-125				C									C			C	C	C	C	C
	-150				C																
XC4002A	-6				CI		CI			CIM											
	-5				C		C			C											
XC4003	-6				CI		CI	MB		CMB											
	-5				C		C			C											
XC4003A	-6				4Q92		4Q92	4Q92		4Q92											
	-5				4Q92		4Q92			4Q92											
XC4004A	-6				4Q92					4Q92				4Q92							
	-5				4Q92					4Q92				4Q92							
XC4005	-6				CI								CIMB	CI	MB						CI
	-5				C								C	C							C
XC4006	-6												4Q92	4Q92							4Q92
	-5												4Q92	4Q92							4Q92
XC4008	-6																		3Q92	3Q92	3Q92
	-5																		3Q92		3Q92
XC4010	-6																		CIMB	MB	CI
	-5																		C		C

C Commercial (0 - 70°C)
I Industrial (-40 - 80°C)
M Military (-55 - 125°C)
B MIL-STD-883, Class B

X2546

Current Software List

The following is a list of the current software revision levels for Xilinx's development system products, as of June 1, 1992.

XCHECKER-PC1 ver. 1.00

XCHECKER-WS ver. 1.00

DS112 Enhanced Serial Configuration PROM Programmer ver. 3.20

DS22-PC1 P-SILOS ver. 4.10

DS290-PC1 VIEWsim ver. 4.13

DS310-PC1 DASH-LCA ver. 4.0

DS343-AP1 Mentor Interface ver.4.02

DS355-PC1 OrCAD VST Pre-Release

DS371 Xilinx ABEL ver. 1.01

DS380 X-BLOX ver. 1.01

DS381 Cadence Design Kit ver. 4.00*

DS390-PC1 VIEWdraw-LCA ver. 4.13

DS391-PC1 VIEWlogic Interface ver.4.13

DS396 XEPLD Workview Library ver. 3.10

XACT 2000/3000 Development System

- **DS501-PC1 ver. 3.20**
- **DS501-SN2 ver. 3.20 on SUN4**
- **DS501-AP1 ver. 3.15 on Apollo**

XACT 2000/3000/4000 Dev. System

- **DS502-PC1 ver. 1.21**
- **DS502-SN2 ver. 1.21 on SUN4**
- **DS502-AP1 ver. 1.11 on Apollo**

* See page 8 for details.

Xilinx EPLD Architecture

Similar in purpose to FPGAs, complex erasable programmable logic devices, commonly referred to as EPLDs, combine the advantages of LSI – smaller size, less cost, higher reliability – with the user's need to create applications-specific circuits without incurring the cost, delay, and risk of mask-programmed gate arrays.

Different from FPGAs, the EPLD architecture is based on programmable logic array technology for both the functional logic and the interconnect structure. Each device contains a number of programmable units, called Function Blocks, each containing nine output macrocells driven by a programmable AND/OR array. A programmable Universal Interconnect Matrix (UIM) routes any device input or any macrocell output to the input of any Function Block, completely eliminating the issue of routability.

This unrestricted programmable interconnect structure, combined with the familiar AND/OR logic of the traditional PAL architecture, makes EPLDs easy to use and easy to understand.

The delay through a Xilinx EPLD device is not only predictable, but also constant. Any function that can be implemented in one pass through the device can run at the maximum specified device speed, 33 or 40 MHz.

Xilinx EPLDs offer two unique advantages over competing EPLD devices.

- The XC7236 and XC7272 contain dedicated high-speed arithmetic carry logic for efficient implementation of fast adders, subtractors, accumulators, and comparators. This overcomes a traditional EPLD weakness.
- The UIM can perform a logic-AND function without additional delay, which means that complex counters of any practical length (32 bits in the XC7236, 64 bits in the XC7272) can run at full speed, even synchronously loadable up/down counters.

No other programmable technology comes close to this performance. Traditional PLDs support no more than 16 bits at full speed, and all channel-routed FPGA devices must concatenate the carry chain, resulting in slower operation for longer counters.

The Xilinx EPLD Data Book provides detailed information on the two available Xilinx EPLDs, the XC7236 and the XC7272.

	XC7236	XC7272
Number of Macrocells	36	72
Number of Function Blocks	4	8
Number of inputs to each Function Block	21	21
Number of product terms per Function Block	57	57
Total number of available product terms	228	456
Maximum number of p-terms available per Macrocell logic function	17	16
Total number of signal pins (input, output, I/O) (largest package)	36	72
Maximum number of pins available for input (largest package)	32	54
Maximum number of pins available for output (largest package)	34	60

XC7236 and XC7272

The XC7272 is a design revision of the original Plus Logic FPGA 2020, while the XC7236 is a design revision of the original Plus Logic Hiper 2010. The product nomenclature was changed to denote the number of macrocells instead of the less relevant gate-count number. As the names imply, the XC7236 has 36 macrocells, and the XC7272 has 72. The XC7236 is the more recent design; it incorporates some feature enhancements beyond the XC7272. The following paragraphs and table describe the extended features of the XC7236 that are supported by the current version of the XEPLD translation software, version 3.1.

- The XC7236 has a direct feedback path from the macrocell output to the OR input of either the same macrocell, or the neighboring macrocell. This speeds up the direct feedback, especially in counters and arithmetic circuits, and it saves UIM connections.
- The XC7236 has one additional FastClock input, for a total of three.
- The XC7236 has selectable logic polarity on all device inputs to the UIM and on device outputs, independent of the feedback.
- The XC7236 allocation of private and shared product terms among the two OR gates feeding the ALU is more efficient; and the ALU function is more streamlined. These differences may result in better functionality, but the software usually isolates the user from such details.

Benchmark Wars

The proliferation of programmable logic manufacturers and architectures has created confusion in the user community. How fast and how dense are these competing devices? How can I compare Xilinx against Actel, or against Altera?

Some vendors have made benchmark claims not only for their own devices, but also for their competitors. This has lead to ridiculously misleading statements: Citing four specific benchmarks, one competitor recently claimed, in public, a two-times speed advantage over the XC4005, when in reality, the XC4005 executes these benchmarks 71% faster. Another competitor has given lengthy comparisons between anti-fuse-based FPGAs, SRAM-based FPGAs, and EPLDs. Most of the "results" were tainted.

Nobody should believe any benchmark claims that are based on one vendor's evaluation of his competitor. A mixture of ignorance and marketing enthusiasm will inevitably distort the results and make them meaningless. Let every manufacturer demonstrate the performance of his devices; leave the comparison to the user.

PREPco, an independent company headed by Stan Baker, known as an editor of Electronic Engineering Times, is coordinating an effort by all PLD manufacturers (including Xilinx) to come up with a set of standardized benchmarks. Each of us will report on our own devices, but all our claims will be verified by our toughest competitor. These benchmarks will, therefore, be accurate and trustworthy. They may not be the answer to every question, and there is room for improvement, for bigger and perhaps more meaningful benchmarks. But we have made an historic beginning. Expect detailed results from PREPco late this year:

Programmable Electronics
Performance Corporation
504 Nino Ave., Los Gatos, CA 95032
Phone: (408) 356-2169
Fax: (408) 356-0195

In the meantime, Xilinx has collected some benchmark data to explain the performance of typical circuits in three different Xilinx architectures: XC3000, XC4000, and XC7200 EPLD.

PA

Xilinx Benchmark Data

		XC7200 EPLD (-25)	XC3000 FPGA (-150)		XC4000 FPGA (-5)	
16-Bit State-skipping Counter, Presettable, non-binary		na	150 MHz	18 CLBs	111 MHz	12 CLBs
16-Bit Binary Counter	Max Speed	40 MHz	116 MHz	24 CLBs	111 MHz	17 CLBs
16-Bit Unidirectional, Loadable Counter	Max Density	40 MHz	20 MHz	16 CLBs	40MHz	8 CLBs
	Max Speed	40 MHz	34 MHz	23 CLBs	42 MHz	9 CLBs
16-Bit Up/Down Counter	Max Density	40 MHz	20 MHz	16 CLBs	40 MHz	8 CLBs
	Max Speed	40 MHz	30 MHz	27 CLBs	40 MHz	8 CLBs
16-Bit Loadable, Up/Down Counter	Max Density	40 MHz	20 MHz	16 CLBs	30 MHz	16 CLBs
	Max Speed	40 MHz	30 MHz	27 CLBs	30 MHz	16 CLBs
16:1 Multiplexer		25 ns	16 ns	8 CLBs	16 ns	5 CLBs
16-Bit Decode from Input Pad		25 ns	15 ns	4 CLBs	8 ns	0 CLBs
24-Bit Accumulator		17 MHz	25 MHz	46 CLBs	32 MHz	13 CLBs
Data Path Benchmark (32 inputs, 4:1 mux, register, 8 bit shift register)		40 MHz	60 MHz	16 CLBs	90 MHz	12 CLBs
Timer/Counter Benchmark (8-bit timer/counter, latch, mux, compare)		40 MHz	30 MHz	21 CLBs	40 MHz	21 CLBs
State-Machine Benchmark (16 states, 40 transitions, 10 inputs, 8 outputs)		40 MHz	—		44 MHz	13 CLBs
Arithmetic Benchmark (4x4 multiplier, 8 bit accumulator)		12 MHz	18 MHz	23 CLBs	18 MHz	21 CLBs
16-Channel, 32-Bit DMA		na	na		20 MHz	72 CLBs

Notes:

1. All speeds are worst-case temperature and voltage.
2. System speeds for slower parts, e.g. XC3000-100, -70, can be approximated by derating appropriately (0.67 for -100, 0.47 for -70).

Ten XC4010 Density Benchmarks

No.	Application	XC4010 Total Gate Count
1	**16-Bit Barrel Shifter or Rotator** 32CLBs i.e. 12 circuits per XC4010 496 gates, all combinatorial, 15.5 gates per CLB	5,952
2	**24-Bit Accumulator** 12 CLBs i.e. 33 circuits per XC4010 583 gates, 168 of them in flip-flops, 48 gates per CLB	19,239
3	**32-Bit Identity Comparator** 9 CLBs i.e. 44 circuits per XC4010 135 gates, all combinatorial, 15 gates per CLB	5,940
4	**9-Bit Parity Checker** 1CLB i.e. 400 circuits per XC4010 28 gates, all combinatorial, 28 gates per CLB	11,200
5	**16-Input Multiplexer** 5 CLBs i.e. 80 circuits per XC4010 31 gates, all combinatorial, 6 gates per CLB	2,480
6	**16-Bit Loadable Counter** 8 CLBs i.e. 50 circuits per XC4010 280 gates, 112 of them in flip-flops, 35 gates per CLB	14,000
7	**100-MHz 24-Bit Programmable Divider** 16 CLBs, i.e. 25 circuits per XC4010 400 gates, 180 of them in flip-flops, 25 gates per CLB	10,000
8	**16 x 8 FIFO** 14 CLBs i.e.28 circuits per XC4010 600 gates, 48 in FF, 512 in RAM, 42 gates per CLB	16,800
9	**32 x 8 Shift Register** 11 CLB i.e. 36 circuits per XC4010 conceptually 1536 gates, all in flip-flops, 139 gates per CLB	55,296
10	**32 x 16 RAM** 16 CLBs i.e. 25 circuits per XC4010 2100 gates, 2048 in RAM, 131 gates per CLB	52,500

The source for these gate count values is the LSI Logic Data Book of July 87:

1. 16-input mux = 31 gates x 16 = 496 gates.
2. 16-bit fast adder (LSI page 3-99) = 277 gates x 1.5 = 415, +24 flip-flops x 7 gates = 583 gates.
3. 8-bit comparator (LSI page 3-74) = 30 gates, x4 = 120 + 4-bit comparator, total: 135 gates.
4. 9-bit parity (LSI page 3-163) = 28 gates
5. 16-input mux = 31 gates.
6. 74161 = 70 gates (LSI page 3-115), x4 = 280 gates.
7. Conservative estimate: less than 1.5 x #6 for same functionality.
8. 128 latches x 4 gates = 512 gates.
9. 256 flip-flops x 6 gates = 1536 gates.

10: 512 latches x 4 bits = 2048 gates plus some addressing.

PA

XC3000-150

Xilinx is currently sampling XC3000 devices in the new -150 speed grade. These devices are 15% – 20% faster than XC3000-125 devices.

The T_{ILO} delay, often used as an LCA benchmark, is reduced from 5.5 to 4.6 ns, with similar reductions in the other timing specifications. As a result, a 16-bit loadable counter, that operates at 28.5 MHz in the XC3000-125, can be clocked at 34 MHz in the XC3000-150, an improvement of 19%.

In new designs, it is sometimes possible to trade the additional speed against CLB usage. For example, in the XC3000-125, a 16-bit carry-lookahead adder settles in 36 ns, while a more complicated and more costly conditional-sum adder requires only 26.5 ns. In the XC3000-150, the settling time of the simple carry-lookahead adder is reduced to 29.5 ns. This provides most of the performance advantage of the XC3000-125 conditional-sum adder, but requires 25% fewer CLBs.

If you would like to simulate your design in an XC3000-150 device, a new speeds file is available through the Xilinx Technical Bulletin Board. The speeds file is the specification data base used by simulators when calculating performance. This file may be downloaded to replace the one currently in your system.

Contact Xilinx for a copy of the XC3000-150 Data Sheet. Production quantities of XC3000-150 devices are expected to be available in September '92.

State Machines Using Xilinx ABEL

The traditional medium for logic design is the schematic diagram. Sometimes, however, the system design process naturally leads to equations or truth tables. In these cases, conversion to gates is an unnecessary extra step. To eliminate it, Xilinx has introduced the Xilinx ABEL software package.

This software package permits blocks of logic to be defined using the ABEL* High-level Design Language. These ABEL definitions can be compiled into LCA netlists without having to represent the logic as gates. An additional benefit is that Xilinx ABEL software optimizes the implementation of the logic to fit the LCA architecture.

State machines are especially suitable for definition by equations or tables. A particular state is entered if, and only if, the current state and the control inputs to the state machine meet a pre-determined set of criteria. Listing these criteria describes the state machine completely.

Given the description of the state machine, the Xilinx ABEL software implements it using a technique that is well-suited to the LCA architecture. While LCA devices provide a large number of flip-flops, the CLB function generators have limited fan-in, and this environment favors One-Hot Encoded (OHE) state machines.

In an OHE state machine, also known as a state-per-bit encoded state machine, one flip-flop is assigned to each state. While fewer flip-flops could be sufficient if states are encoded, flip-flips are not normally a critical resource in LCA designs; the critical resource is CLB function-generator inputs.

OHE minimizes the complexity of the next-state logic associated with each flip-flop by spreading the task across the larger number of flip-flops. With OHE, a flip-flop is only set when its specific state is entered. Identifying the current state by a single bit makes it easier to combine the current state information with the control inputs.

State machines using OHE are typically faster than those using conventional state encoding. The reduced logic complexity results in fewer levels of CLBs to define the operation of each flip-flop. Consequently, logic delays are lower, and clock rates can be higher.

The inexpensive Xilinx ABEL software is not limited to state machine design. The ABEL language can be used to define any logic that is more conveniently defined using equations rather than gates.

BN

* ABEL is a trademark of Data I/O Corp.

X-BLOX Provides High-Level Schematic Entry

Have you ever sketched a block diagram on the back of an envelope, and wondered how well it would work? X-BLOX tools make it easy to find out. With the X-BLOX synthesis design tools, you can enter your block diagram into the existing design-entry package, and then have it automatically implemented in an XC4000 LCA device.

Using X-BLOX, a library of 30 frequently used block-diagram functions are available to construct designs. This library contains registers, adders, and counters; even RAMs and ROMs are provided. If necessary these functions can be combined with gate-level logic and any other library elements that are available.

The X-BLOX library is not just a collection of macros, however. A rule-based expert system converts each X-BLOX module in your design into an optimized custom Hard Macro. These Hard Macros are implemented in a way that best exploits the LCA architecture.

Wherever possible, the Hard Macros utilize the advanced features of XC4000 LCA devices. All adder and counter macros exploit the dedicated carry logic for maximum performance and minimum CLB count. RAM and ROM macros are automatically created by the X-BLOX software, and MEMGEN is not required.

Hard Macros improve the performance of the design by carrying the functional structure of the design into the implementation phase. CLBs that are common to a particular function are kept together in the array. This improved placement enhances both the routability and the performance.

X-BLOX tools make it easy to change designs. Function modules are interconnected by single-line busses that are easily modified. Consequently, adding or removing modules is a simple task.

Bus widths are controlled by a single parameter located anywhere in a data path. Editing this one parameter not only changes the bus width throughout the data path, but also changes the width of all the functional blocks in the path. Even RAMs automatically change their depth to match the number of address bits.

High-level design with the X-BLOX design tools shortens the design cycle without sacrificing performance. Your engineering productivity is increased; more importantly, your product goes to market sooner.

BN

New Applications Data Base

In late January, we installed a call tracking and problem resolution database for the Technical Support Hotline. Instead of having to keep individual notes about phone calls, our Applications Engineers (AEs) now have quick and easy access to a central record of customers, technical questions, and known problems.

When you call the Hotline, our Customer Response Center (CRC) asks you for your name and then calls up your record. After your first call, you never have to tell us again the details of your company address and phone number, the type of Xilinx software that you have installed, and the computer it runs on.

Attached to your record are files for each of your previous calls. Each call record includes information about: when the call was opened, the Applications Engineer to whom it was referred, the topic of your question or request, notes about your questions, whether the record is open or closed, and when it was closed. The CRC staff can see your entire call history at once, and refer your call to the Applications Engineer best able to answer your question.

The AE can browse through your call history and see what you have discussed with other engineers during previous calls to the Hotline.

If no AE is available to take your call, the CRC staff queues the call so that the next available AE can accept it and call you back. Similarly, if you want to leave a message for an engineer who is not on phone duty that day, the CRC staff will queue it to the AE's message bin.

The database is not only for call tracking, it also contains records of common problems and their solutions. Using symptom keywords from your description of the problem, the AEs can search the database, using the captured expertise of our entire Applications group.

In the future, we plan to make this part of the database available to our users. You'll be able to browse through the Customer Access Database and either download or fax back to yourself detailed explanations of known problems. Look for it next year.

DF

ADI Version 3.20 Now Shipping

Xilinx has just released the latest version of the ADI software used with XC2000 and XC3000 LCA devices. Many improvements have been made resulting in more demanding designs being routed automatically.

The new algorithm significantly improves both the density and the performance of almost any design. Using the new algorithm, most designs that failed to route with the old algorithm are now completed automatically.

When compared using the Xilinx rogues'gallery of difficult designs, ADI version 3.20 completely routed more designs than before, and provided implementations that operated faster. Specifically designed to route LCA devices, the new router outperforms third-party tools.

The new router uses net delays to direct its operation. Nets that are routed early have greater access to routing resources. If nets that are routed later become excessively slow, previously routed nets can be "ripped up" to accommodate them. Similarly, previously routed nets can be modified to accommodate those that cannot be routed with the remaining resources.

Flagnet and Weightnet constraints should no longer be necessary with the new router, and may even be counter-productive. Flagged nets are routed first, and cannot be ripped up. Consequently, the resources they use cannot be re-allocated by the router, and slow or unrouted nets may result.

Along with the new routing algorithm, there is a new placement algorithm. The new algorithm operates much faster than the old one, yet provides results that are almost as good. Selecting the new algorithm with the -Y option reduces the LCA compilation time, and thereby increases design productivity.

Support for 3-state busses is greatly improved. TBUFs with a common 3-state control are automatically aligned in a column. This permits a vertical Longline to be used for the 3-state control. In addition, logic associated with the TBUFs is placed close to this column, improving both performance and routability.

TBUFs can also be named, using the BLKNM attribute. Naming TBUFs permits them to be referred to easily in constraints files, improving the user's ability to floorplan within the CLB array.

To improve the floorplanning of inputs and outputs, IOBs can now be constrained to one edge of the device, or to half an edge. This constraint is often adequate to simplify PCB layout. However, it has a much smaller impact on LCA routing than locking a signal to a specific pin.

Fully Integrated Design Environment from Cadence

Under a new OEM agreements, Cadence Design Systems will integrate the Xilinx XACT development system into the Composer* design environment. This combination will provide a complete top-down-design environment and the convenience of a single software vendor, since the complete package will be sold and supported by Cadence.

In addition to entering designs as schematics, high-level design languages such as VERILOG-XL* and VHDL-XL* will be available to designers. X-BLOX will also be integrated into the design environment.

The top-down-design process will be timing-driven, with board-level timing objectives being passed to the FPGA implementation software. Timing information from the resulting FPGA design will then be passed back to the board-level model for simulation.

This closed-loop approach permits systems designers greater visibility into their designs, and greater control over the design process. Consequently, system requirements will be met more reliably, and in less time.

All LCA families are to be supported, and FPGA design kits will be available from Cadence, starting in the third quarter of '92. Initially, the software will run on Sun workstations, with other platforms to follow. Valid design kits will also be available.

* Composer, VERILOG-XL and VHDL-XL are trademarks of Cadence Design Systems, Inc.

Enhancements to PPR

Version 1.20 of the XC4000 partition, place and route software (PPR) is now available. This version offers designers several new features, aimed at improving performance and increasing productivity. Partitioning, placement and routinghave been enhanced.

XC4000 designers can now control logic partitioning at the schematic level. New FMAPs and HMAPs operate similarly to CLBMAPs in XC3000; the user can specify how gates are combined into CLB function generators.

In critical paths, the logic partitioning affects the performance of a design. FMAPs and HMAPs force PPR to use the partitioning envisioned by the designer. Consequently, performance requirements are more predictable.

The placement process, which often has the greatest effect on performance, is initialized by a random seed value. This component of randomness causes repeated runs of PPR to result in different placements. The new version of PPR can automatically evaluate the placements resulting from a user-defined number of seed values, and choose the best.

A new routing feature also helps ensure that performance requirements are met. Version 1.20 permits a maximum net delay to be specified. PPR continues rerouting the design until the longest net delay is less than this maximum value.

If PPR is unable to meet the specification, after a number of attempts determined by the user, it tries to meet a second, less demanding specification. If this requirement is also unattainable, PPR increases the delay specification in fixed increments until the requirement can be met, or until the delay exceeds a worst-delay specification. In the latter case, PPR leaves unrouted those nets that do not meet the specification. All these delay specifications and increments are user definable.

Other features of version 1.20 include better support for the wide edge decoders, and improved constraint handling. The new version also provides better support for back annotation of the schematic.

Control XC4000 Placement with Hard Macros

Hard Macros in XC4000 are often associated with adders and counters that use the dedicated carry. However, Hard Macros can improve the performance of most functions that benefit from controlled placement for efficient routing. To supplement the XC4000 Hard Macro Library, which contains many common functions, the program HMGEN lets the user create his own Hard Macros.

Unlike Soft Macros that define only the logic function, a Hard Macro contains additional information that defines relative CLB placement and pin utilization, and may also include routing information. However, it is easy for the autorouter to provide efficient routing once a good CLB placement has been defined.

Once constructed, Hard Macros are used in schematics like any other macro. PPR places a Hard Macros in the CLB array, according to the needs of the other logic. However, within a Hard Macro, relative CLB locations are not changed, and the routing advantages designed into the macro can be exploited by PPR.

To create a non-arithmetic Hard Macro, the logic schematic of the macro is entered normally, using FMAPs and HMAPs to determine the logic partitioning. PPR is then used to create an LCA file that is edited in the XACT Design Editor (XDE) to provide the desired placement. Any input and output pads are deleted, and the design is unrouted. After design rule checking (DRC), the resulting LCA file can be converted into a Hard Macro using HMGEN. Arithmetic Hard Macros are best created by modifying an existing Hard Macro in XDE.

The creation of user-defined Hard Macros is not recommended for inexperienced designers. Besides requiring use of the XDE, defining a Hard Macro requires that the designer consider the impact of using the macro in the LCA device. Inputs and outputs must be positioned within the macro such that they communicate efficiently with surrounding logic; the effect the macro has on over-all routing must also be considered. As a further constraint, Hard Macros must be rectangular in shape; unused resources within the rectangle are not available to PPR for other logic.

HMGEN is not part of any released Xilinx product but is available free of charge from Xilinx.

BN

XC4000 Dedicated Carry Logic

XC4000-series CLBs contain dedicated, hard-wired carry logic to accelerate and condense arithmetic functions such as adders and counters. Adders achieve carry delays as low as 750 ps/bit, while utilizing only half a CLB/bit. This is certainly denser than any other approach, and in most cases, faster.

The dedicated carry logic uses a simple ripple scheme for maximum flexibility. Adders and counters may be of any length, start anywhere in a column of CLBs, and have their MSB at either the top or the bottom. The carry can run up or down a column of CLBs, and can also run sideways at the top and bottom, accommodating very long adders and counters; when the adder or counter reaches the top or bottom of a column, it simply turns around and continues in the next one, and with no loss of performance.

Only the carry path of the adder is implemented in dedicated logic, as shown in the figure. Between CLBs, the carry is routed on special interconnect lines that are only available to the carry logic. This combination of dedicated logic and high-speed interconnect provides the high carry-propagation speed.

The carry network supplements the other resources in the CLB. The outputs from the carry chain are available as inputs to the CLB function generators. The adder sums are formed in these function generators, just as any other logic function.

In addition to the basic adder configuration, the carry logic and function generators may be configured to provide a subtracter or an adder/subtracter. All three functions can be modified to work with a single operand, providing incrementers and decrementers. These incrementers and decrementers, together with the CLB flip-flops are used to generate the high-speed counters.

The performance achieved by the dedicated carry logic is outstanding; 16-bit adders and subtracters settle in 20.5 ns, and yet consume only eight CLBs. 32-bit adders and subtracters use 16 CLBs, and settle in 32.5 ns.

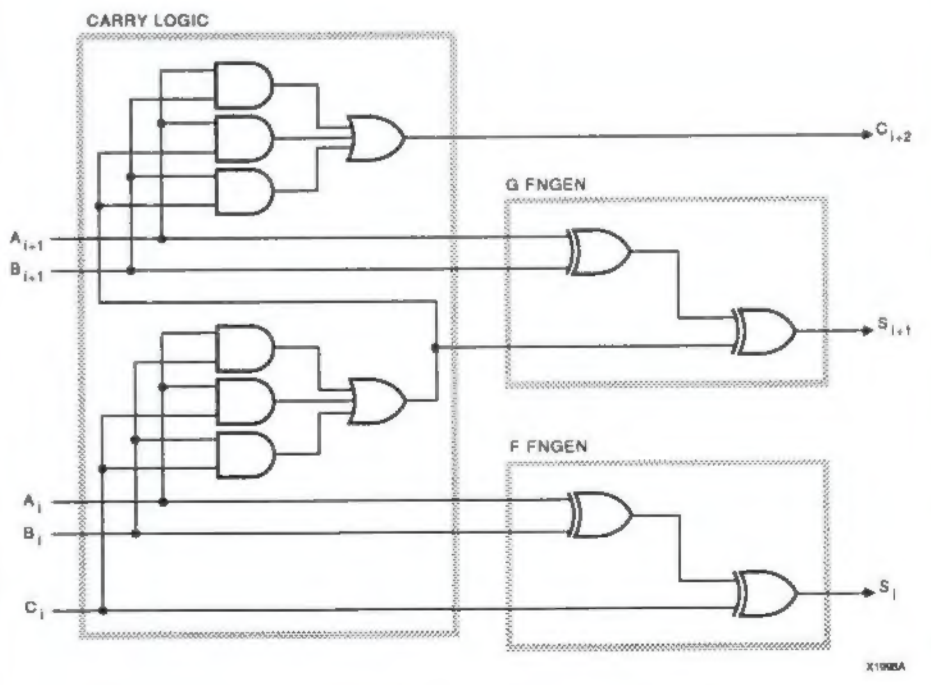
Loadable up counters and down counters use the same number of CLBs, and support clock frequencies of 40 MHz for 16 bits and 27 MHz for 32 bits. Non-loadable up/down counters also achieve these speeds and CLB counts, while loadable up/down counters are slightly slower, and required additional CLBs.

Currently, the user can access the dedicated carry logic in three ways. Firstly, Xilinx provides a library of Hard Macro adders and counters that use the carry logic. These Hard Macros may be incorporated directly into user designs at the schematic level. Alternatively, the user may generate his own Hard Macros using the HMGEN utility.

The third way to access the dedicated carry logic is through X-BLOX. All counters and arithmetic functions in X-BLOX are implemented with the carry logic. A Hard Macro of the appropriate size and functionality is automatically generated and used in the design.

In the future, it will be possible to access the dedicated carry logic at the schematic level.

For more information on the dedicated carry logic, please refer to the Xilinx Application Note *Using the Dedicated Carry Logic in XC4000* (XAPP 0013).



Conceptual Diagram of a Typical Addition (2 Bits/CLB)

Estimating Adder and Counter Performance

In most LCA designs, performance cannot be estimated with any accuracy until after implementation. This is because the performance is affected by routing delays, and, prior to implementation, these are not known. However, in adders and counters using the XC4000 dedicated carry logic, delay estimation is possible.

The carry path in an adder uses dedicated interconnects between CLBs. These interconnects introduce a fixed delay, even when the carry passes from one CLB column to the next at the top or bottom of the array. This permits the routing delay to be incorporated into the CLB specifications published in the data sheet. As a result, the propagation delay through an adder can be calculated using only data-sheet specifications.

For a typical adder, this calculation can be reduced to a simple formula. In an XC4000-5, the maximum propagation delay from the operand input to the sum output of an N-bit adder is approximately

$$t_{pd} = 8.5 + 0.75\,N \quad \text{ns}$$

This estimate does not include the delay from the operand source register to the adder or any additional delay reaching the destination register. However, it is still a useful benchmark.

For an N-bit counter, the minimum clock period that permits the carry path time to settle is approximately

$$t_{clk\text{-}clk} = 13 + 0.75\,N \quad \text{ns}$$

For more information on the derivation and limitations of these formulae, please refer to the Xilinx Application Note *Estimating the Performance of XC4000 Adders and Counters* (XAPP 018).

BN

Lower Cost for High-Volume Production

LCA devices are recognized as a cost-effective means of implementing logic during development and for limited volume production. However, for higher volume production, it may be necessary to reduce component costs. Masked gate arrays are an option, but converting a design to a gate array is costly in both time and money.

For mid-volume production, HardWire™ devices offer reduced component cost, avoiding the high cost of conversion because the mask-programmed HardWire devices are architecturally identical to their RAM-programmed counterparts. All the effort put into implementing the original design is re-used.

The HardWire mask is derived from the routed LCA file. Consequently, the HardWire device is guaranteed to be logically correct. The HardWire device is also guaranteed to meet, or beat, the worst-case delays of the RAM-based design. Logic partitioning, CLB locations and routing are all unchanged by the conversion, and mask-programmed interconnections are always faster.

Compare this automated, low-risk approach to the gate-array conversion process: first, the netlist must be converted to gate-array format, followed by placement, routing, and simulation to verify the timing. These steps may need to be iterated several times to satisfy a critical timing requirement.

While gate arrays require a new set of test vectors to be developed for each design, Xilinx automatically generates test vectors for HardWire devices. Using dedicated scan-test logic built into the device, these test vectors provide 100% fault coverage.

HardWire device benefits go beyond the ease of conversion. An unexpected increase in the demand for a product that uses HardWire devices can easily be accommodated. Simply revert temporarily to the readily available programmable version that fits the same socket. Production is not delayed by complete dependence on a long-lead-time custom product.

RAM-based devices also simplify the addition of new features to extend the life of a product. Develop them in a RAM-programmable device in the production board and move the new design immediately into production, using RAM-programmed devices until a new HardWire device is available.

As production typically slows at the end of the product life cycle, it may again be more cost-effective to use RAM-programmed devices. Production is not constrained by minimum purchase requirements, and the standard devices will continue to be available as replacement parts.

Virtually all LCA designs are suitable for HardWire conversion. The only restriction is that correct operation may not depend upon some minimum delay; the HardWire device can be much faster. Such asynchronous design is considered bad practice, and a special design-rule-checking (DRC) program is used to evaluate all designs prior to conversion and to flag potential problems.

The first step in converting an LCA design to a HardWire design is to submit the design for evaluation. Once it is approved, prototype HardWire devices are available in six weeks. Production quantities follow eight weeks later.

HardWire versions of all three LCA families are available.

Boundary Scan Simplifies Board Test

Testing of printed-circuit boards can be a major problem. While integrated circuits can be tested before insertion, there is no way to obtain such a high level of confidence in the interconnection network on which they depend. Simple open circuits and short circuits in the interconnect are fatal flaws, but are often difficult to detect.

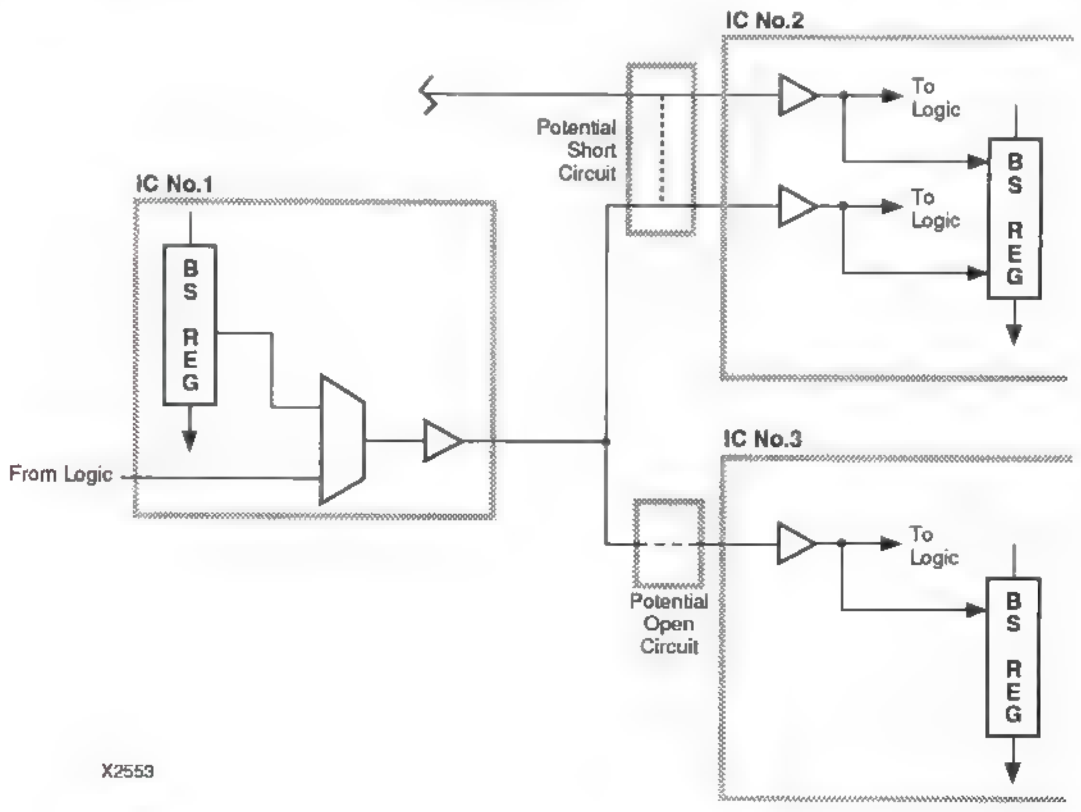
Connectivity testing is a significant problem. As ICs become more complex, it is increasing difficult to control and observe the printed circuit traces between them. Probing the card with a "bed-of-nails" tester offers a possible solution, but, with denser packaging and devices surface-mounted to both sides of the board, even this becomes problematic.

Simplifying printed-circuit-board connectivity testing is the primary purpose of boundary-scan diagnostics. Circuitry associated with each pin permits a known signal to be driven onto every trace, and the signal to be checked at every destination. This test can be repeated with different signals to detect open and short circuits, and this can be done for every driver.

Test vectors are distributed to the drivers through a serial transmission path. The same path, which includes all inputs and outputs, is used to recover test results. The serial transmission scheme is standardized so that different vendors' products can work together. Its operation is defined by IEEE specification 1149.1, sometimes referred to as the JTAG specification after the committee that originated it.

IEEE 1149.1/JTAG boundary-scan testing is supported in XC4000-series LCA devices. Dedicated logic and pins provide a Test Access Port (TAP), as defined by the standard, and dedicated logic in the IOBs provide the boundary-scan Data Register and associated test functions.

Built-in self-test of the LCA device, which is optional under the IEEE specification, is not explicitly supported. However, the XC4000 LCA architecture permits internal logic to be connected to the TAP. Test logic can now be configured into the LCA device, and its operation controlled by the boundary-scan test system.



Boundary-Scan Diagnostics

Implementing Boundary Scan in XC3000

Although XC3000-series LCA devices do not contain dedicated boundary-scan logic, it is possible to configure an XC3000 to emulate boundary scan. This emulation consumes a significant amount of the LCA resources (almost all in an XC3020), and it is not suggested that boundary scan be built into a working design. However, because the RAM-based LCA device is reconfigurable, it can be configured for board testing, and then reconfigured for operation.

Four pins must be dedicated to the Test Access Port (TAP). Due to external interconnection requirements, these pins can probably not be re-used in the actual design. The TAP Controller, Instruction Register, Bypass Register and Test Data Output Buffer together with miscellaneous logic require 11 CLBs.

The CLB requirement for the Test Data Register depends upon the number of IOBs used, and how they are configured. Each requires between 0.5 and 1.5 CLBs, plus one CLB for each distinct 3-state control. While this may not allow every IOB to be bidirectional with an independent 3-state control, it will accommodate most designs.

A specific boundary-scan emulation must be created for each LCA design. This comprises the 11 CLBs of core logic, which is common to all emulations, and a Test Data Register concatenated from four macros according to the output usage in the design.

For more information on using boundary scan in the XC3000, see the Xilinx Application Note *Boundary Scan Emulator for XC3000* (XAPP 007).

BN

LCA Performance: Ask the Right Question

Before starting an LCA™ design, it is a good idea to do some quick performance calculations, just to make sure you are in the right ballpark. It is tempting to try estimating the highest speed that the design can achieve. However, it is usually much easier, and just as useful, to determine whether a predetermined speed can be attained.

Given the desired clock frequency, it is easy to estimate the logic complexity that can be supported. This complexity can then be compared to the functional requirements to determine feasibility. Only in marginal cases is a complete speed evaluation necessary.

Typically, a data path runs from a register, through some combinatorial logic to another register. In an LCA device, the shortest data path involves a CLB clock-to -output delay plus a CLB set-up time: a total of 9.5 ns in an XC3000-150. However, this time does not include any allowance for routing. Adding 4 ns for routing, the shortest data path is typically 13.5 ns.

If additional combinatorial CLBs are added into the data path, each level of CLB adds 4.5 ns, and additional routing delay is also introduced. Including a typical routing allowance, 8.5 ns should be added for each level of combinatorial CLB.

This simple speed-estimating procedure can also be reversed. If the system clock frequency is 30 MHz, the 33-ns period typically provides for two combinatorial CLBs between registered CLBs.

Clock period	33	ns
Minimum delay	13.5	ns
Remaining	19.5	ns
Each combin. delay	8.5	ns
# of combin. CLBs possible	2	

Including the function generator in the destination CLB, a total of three function generators can be cascaded. Knowing the number of function generators that can be cascaded, the design can be analyzed to determine whether or not it is feasible.

Of course, this is only a very rough calculation intended to establish feasibility; it neither establishes a performance limit, or guarantees that a level of performance can be achieved. It does, however, give some indication of the level of difficulty involved in the design.

In addition, critical areas can be identified prior to starting the design. It is better to accommodate critical areas from the outset, rather than "fix" them later. Conversely, if a design only requires a fraction of the capability available, it might be possible to multiplex some functions, and provide a less costly solution.

For information on how to use this procedure in other LCA devices, see the Xilinx Application Note *LCA Speed Estimation: Asking the Right Question* (XAPP 011).

BN

Faster Multiplexers in XC3000

The traditional building block for large multiplexers in XC3000 is a dual 2-input MUX. This building block comprises two functions of three variables, and uses all five inputs to the CLB. A 4-input MUX cannot be constructed in a single CLB since it requires six inputs.

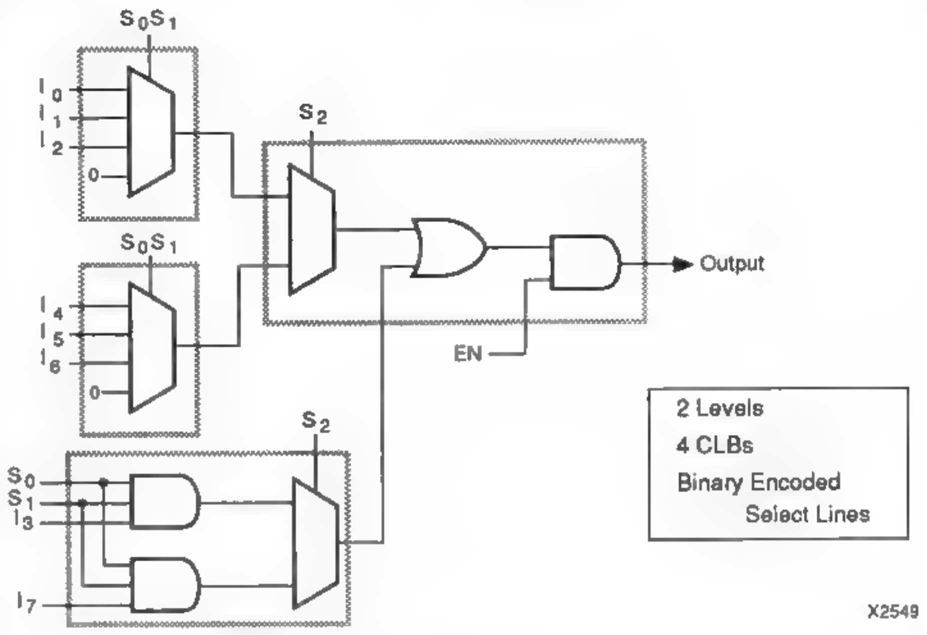
Using the dual 2-input MUX, larger multiplexers can be constructed using a conventional tree approach, with each select bit associated with one CLB level. This results in 8:1 multiplexers that use four CLBs in three levels, and 16:1 multiplexers that use eight CLBs in four levels.

However, a 3-input MUX can be implemented in only one CLB. Such 3-input MUXs can implement larger multiplexers that have less delay, while retaining the binary encoding of the select lines.

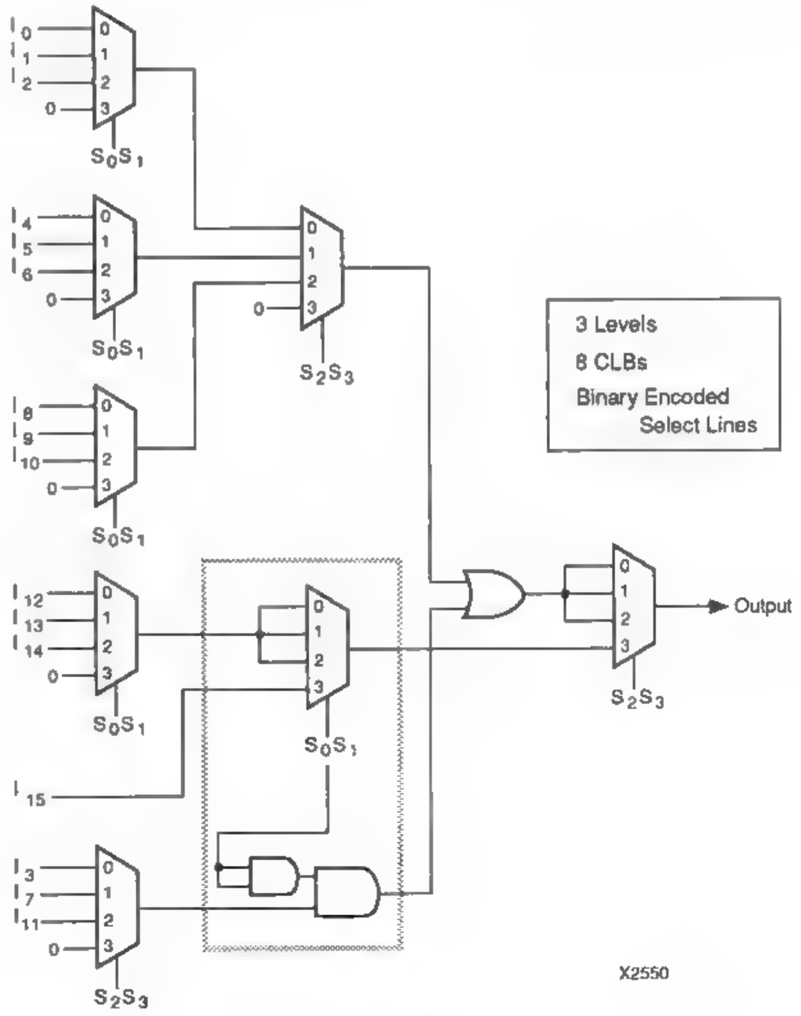
The 8:1 multiplexer, shown below, also provides an enable input. Again, four CLBs are used, but with only two levels of delay. The enable input permits the multiplexer to be expanded using only one additional level of CLBs. Decoded select lines are used to enable up to five 8:1 multiplexers into an OR gate. In this way, 3-level multiplexers with up to 40 inputs may be constructed.

For 16:1 multiplexers, the second design uses eight CLBs, and again has three levels of delay. It also has binary-coded inputs, and uses fewer CLBs than two 8:1 multiplexers with the necessary expansion logic.

BN



8:1 Multiplexer



16:1 Multiplexer

XC3000 Counters

When selecting a counter design for a specific application, there are three primary considerations; does it meet the functional requirements, is it fast enough and could it use fewer LCA resources?

The functional requirements that must be considered include binary/non-binary operation, up, down and up/down counting, loadability, the provision of set/clear, count enable, and synchronous operation to permit output decoding. Speed and resource utilization are self-explanatory, and can often be traded against each other.

However, it must be realized that as a counter becomes more complex, it usually becomes both larger and slower. The table summarizes the characteristics of various counter designs available for the XC3000.

For a more detailed description of the designs mentioned below, see the individual Application Notes.

High-Speed Synchronous Prescaler Counter (XAPP 001)

This simple design provides a very basic non-loadable, up counter with a count-enable control. However, this simplicity permits it to be both the densest and the second fastest design. It is easy to convert the design into a down counter, but not possible to convert it into an up/down counter.

Simple, Loadable, Up/Down Counter (XAPP 002)

Being loadable, this counter is unable to benefit from the prescaler technique, and a simple ripple-carry scheme is used throughout. Consequently, it is slower than the above design. The maximum clock frequency is inversely proportional to the length of the counter; the ripple-carry path incurs one T_{ILO} delay for each two bits.

A modification to this counter almost doubles the maximum clock rate by dividing the carry path into two halves. With this modification, the carry path settles in approximately half the time. However, this modification requires one additional CLB.

Synchronous Presettable Counter (XAPP 003)

In this design, speed is increased by replacing the serial gating of the ripple-carry path with parallel gating. Ideally, with arbitrarily wide gates, the carry-path settling time could be reduced to one gate delay.

However, with limited gate width, the settling time increases logarithmically with counter length; this is still a significant improvement over the linear increase seen previously, especially in longer counters. The additional speed is achieved at the cost of using more CLBs with more complex routing.

Loadable Binary Counter (XAPP 004)

The loadable binary counter also uses parallel gating to accelerate the carry path. In this case, however, a more structured approach is taken. A fast lookahead-carry technique is used, resulting in a carry path with a consistent depth of gating. Consequently, there are many equally critical paths.

The regular structure lends itself to hand placement for maximum speed. The irregularity and smaller number of critical paths of XAPP 003 reduces its dependence on CLB placement, benefiting the automatic placement tools. XAPP 003 performance may be improved by re-routing a few critical paths, but it will not match an optimally placed XAPP 004.

Ultra-Fast Synchronous Counters (XAPP 014)

In some applications, such as clock division, the only requirement is a high clock rate. This counter is designed to fill that need. It is approximately twice as fast as XAPP 001 described above, but uses almost twice as many CLBs.

The key is the use of a prescaler technique, together with an active Longline to distribute the parallel count enable. This distribution scheme uses replicated flip-flops to eliminate delay but depends upon the predictability of the binary sequence.

Counter Performance in XC3000-150

	Loadable	Up	Down	Up/	8-Bit		10-Bit		12-Bit		16-Bit		20-Bit		24-Bit		32-Bit	
				Down	MHz	CLBs	MHz	CLBs	MHz	CLBs	MHz	CLBs	MHz	CLBs	MHz	CLBs	MHz	CLBs
XAPP 001		•			80	5	68	8	65	9	65	14	63	17	63	21		
XAPP 002	•	•	•	•	31	8	25	10	25	12	20	16	15	20	13	24		
XAPP 002	•	•	•	•							28	17						
XAPP 003	•	•	•		39	8			32	15	31	20						
XAPP 004		•	•								34	23					23	49
XAPP 004				•							30	27					23	56
XAPP 014		•									116	24						

150-MHz Presettable Counter in XC3000

Prescaling is an established technique for high-speed counters. Using a derivative of this technique, LCA devices can implement a presettable counter at the full 150-MHz toggle rate of an XC3000-150. These counters can be up to 24-bits long.

In a prescaler counter, a small, very fast counter divides the clock rate. The divided clock is provided to a large, slower counter that is unable to settle at the fast clock rate. However, even when implemented synchronously, a conventional prescaler counter cannot be loaded; the technique depends upon the predictable binary sequence to ensure that the larger counter has adequate settling time.

If the prescaler counter is loaded with an arbitrary value, the binary sequence is broken, and the settling time of the larger counter is no longer guaranteed. To ensure an adequate settling time, either the clock frequency must be reduced significantly, or the values that can be loaded must be severely restricted.

To provide presettable prescaler counters, John Nichols of Fairchild Applications introduced a pulse-swallowing technique in 1970. It uses a dual-modulo prescaler that can divide the clock by 2^n or 2^n+1. See page 6-38 of the Xilinx 1992 Data Book for further information.

Twenty years later, Xilinx developed a variation of the the pulse-swallowing technique for use in LCA devices. This technique, called state-skipping, uses a dual-modulo prescaler that can divide by 2^n or 2^n–1.

In a state-skipping counter, the prescaler is not loaded. Instead, the least significant bits of the load value are used to initiate a correction counter that controls the modulus of the prescaler. Consequently, the larger counter, that contains the more significant bits, always has at least 2^n–1 clock periods in which to settle, even after a load.

Typically, the minimum of 2^n–1 clock periods between the load and the first clock to the larger counter is longer than is required. To compensate, the prescaler operates with its shorter cycle until any extra delay has been nullified. This compensation is controlled automatically by the correction counter.

For example, in a counter using ÷7/÷8 prescaler, the value loaded might require the first clock to the larger counter occur 5 clock periods after the load. In this case, the minimum 7-clock cycle period of the prescaler delays the first clock to the larger counter by two periods.

To nullify this extra delay, the prescaler continues dividing by 7 for a further two cycles, cancelling one clock period of the extra delay each cycle. The third clock to the larger counter occurs 21 periods after the load, which is the same as in a conventional counter (5 + 8 + 8 = 21 clocks). Once the compensation is complete, the prescaler returns to dividing by 8.

Clearly, the counter will operate in a non-binary manner while the correction is being made. During this time, the counter skips a state each cycle of the prescaler, hence the name of the technique. The maximum time to complete the correction is 2^n–1 cycles of the prescaler. A further consequence of state-skipping is that some small division ratios cannot be used, because the correction cannot be completed within the period of the counter.In addition, the load must be synchronized with the prescaler cycle. This happens automatically if the counter is loaded when it reaches TC. This is common practice for timers and dividers, which are excellent application for state-skipping counters.

With these exceptions, a state-skipping counter may be loaded exactly like a conventional binary counter. There is no need to modify the load value required for any given divide ratio, as is necessary with a pulse-swallowing counter.

One advantage of the state-skipping technique that is peculiar to LCA implementation, is that a ÷3/÷4 prescaler can be built in a single CLB. This is the key to the 150-MHz presettable counter, shown in the Figure.

The counter uses two state-skipping prescalers in cascade. Each is a 2-bit dual-modulo prescaler that divides by 3 or 4, and each has its own correction counter. Only the first prescaler is clocked by the high-speed clock. The maximum clock rate to the remainder of the counter is at least three times slower.

The first prescaler is implemented in a single CLB, and the counter design allows the control inputs several clock cycles to set up. Consequently, the high-speed clock is limited only by the toggle rate of the flip-flops in this CLB. In an XC3000-150 this is 150 MHz.

The remaining counters, including the first correction counter, are all clocked by Q_1. This synchronous operation permits the correction counters and $Q_4 - Q_{23}$ to be loaded by Terminal Count in a conventional way.

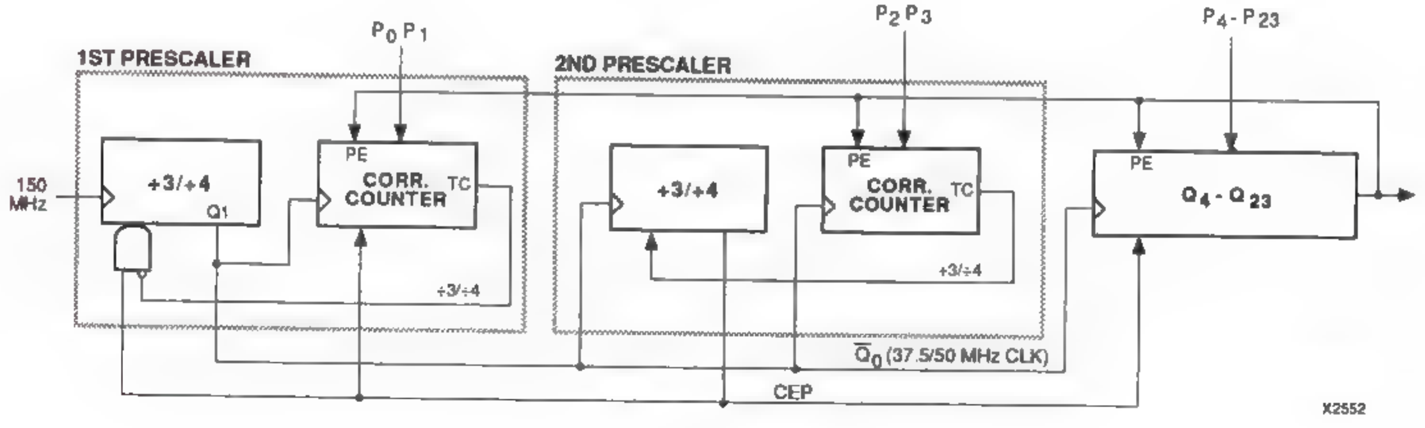
In each cycle of the second prescaler, only one of the three or four first-prescaler cycles can be a correction cycle. Consequently, the the divide ratios of the composite prescaler are limited 11, 12, 15 and 16, depending on which prescalers are correcting. This permits the $Q_4 - Q_{23}$ counter at least 11 clock cycles in which to settle, and distribute the parallel enable signal.

Each time a prescaler correction cycle occurs, the corresponding correction counter is decremented. Correction cycles continue while the correction counters are non-zero. When zero is reached in either of the correction counters, the corresponding prescaler ceases correcting, and that correction counter remains at zero until it is reloaded.

Correction can take up to 45 clock periods to complete, and during this time some counter values will be skipped. However, the counter behaves in a conventional binary manner after less than 46 clock cycles. Some divide ratios below 30 cannot be used, since the correction time is greater than the counter period, but all divide ratios of 30 or greater are available.

State-skipping counters are the subject of an upcoming series of Applications Notes. Design files for the 24-bit 150-MHz Presettable Counter are available as XAPP 021.

BN



150-MHz Counter

Accelerating XC4000 Counters

The dedicated carry logic in XC4000 LCA devices provides a mechanism for very fast and efficient counters. While the ripple-carry scheme appears simplistic, the hardware implementation of the dedicated carry logic is very fast, and requires few CLBs. In fact, the implementation is so efficient that it defeats most attempts to replace it. It is possible, however, to augment the operation of the carry logic and obtain higher performance.

To accelerate the counter, the effective length of the carry path must be shortened. This is achieved by dividing the counter into two sections that settle in parallel, as shown in the Figure. The carry output of the less significant section provides a parallel Count Enable (CEP) to the more significant section. The use of CEP is most often associated with prescaler operation, but this is not necessarily the case.

In a prescaler counter, CEP is typically decoded from the least significant two or three bits. The CEP signal is then used to enable the remaining bits, such that their effective clock rate is one fourth or one eighth of the actual clock rate. This allows multiple clock periods for the remaining bits to settle; the whole counter can be operated at the speed of the prescaler, in spite of the long carry path.

Using the prescaler technique, however, it is not possible to load the counter and guarantee that it will count correctly on the following clock cycle. The carry chain in the more significant bits is designed to settle in multiple clock periods. If these bits are enabled to count on the clock following the load operation, the carry path will not, in general, have had adequate settling time. Depending on the value loaded, it might not be possible to resume counting for several clock periods after the load operation.

This problem may be avoided if the carry chain is be divided into approximately equal halves, both of which can settle within the clock period. The parallelism in the carry chain reduces the clock period, but not as dramatically as with a prescaler. However, loadability is retained.

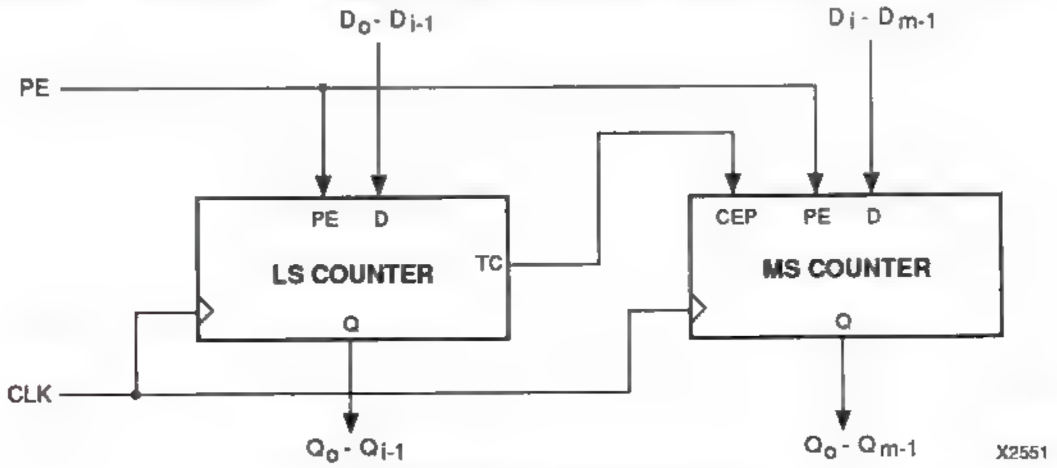
The carry delay is reduced to the settling time of the more significant section of the counter, or the settling time of the less significant section plus the subsequent routing and count enable times, whichever is greater. For optimum performance, the counter must be divided into unequal halves such that these times are balanced.

This technique is most effective in long counters. For example, a 32-bit counter in an XC4000-5 can be accelerated from 27 to 34 MHz. Variations of the technique, however, permit some advantage to be gained in counters as short as six bits.

In non-loadable counters where the prescaler technique is used, the critical delay is usually the distribution of CEP. Using a 1-bit prescaler, this delay can be reduced by replicating the prescaler (the LSB of the counter), such that everywhere CEP is used, it is available from the adjacent CLB. This in effect creates an "active Longline." Adding a second prescaler stage permits this technique to be used in significantly long counters.

The active Longline technique is expensive in CLBs, but boosts the clock frequency to the full shift-register frequency of the LCA device. In an XC4000-5, this is 110 MHz.

For more information on these counters, see the Xilinx Application Notes *Accelerating Loadable Counters in XC4000* (XAPP 023), to be published shortly, and *Ultra Fast Synchronous Counters* (XAPP 014).



Accelerated N-Bit Counter

Thin Quad Flat Pack (TQFP-100)

Xilinx offers a new, smaller outline 100-pin package option for the XC2000 and XC3000 families. The package body dimensions are 14 mm x 14 mm x 1.4 mm (0.55" x 0.55 "x 0.055") with 25 gull-wing leads on each side. Lead pitch is 0.5 mm (0.02") and the total PC-board footprint area is only 16 mm x 16 mm (0.63" x 0.63").

The XC3042 in a TQFP-100 offers 3000 user-programmable gates in the space of a dime. (The diameter of a dime is 18 mm, its thickness is 1.4 mm).

This package is ideally suited for PC Card and other high-density applications. It is also an ideal match for the low power consumption and in-situ programmability of the Xilinx FPGAs. Anti-fuse or EPROM-based programmable devices cannot use such a fine-lead package, because the insertion into a programmer would inevitably bend the leads out of alignment.

XC2018, XC3030, and XC3042 devices in TQFP are now shipping in production volume.

H-Spice Models Are Available

H-Spice models of Xilinx LCA output circuits are available from:

Meta-Software
1300 White Oaks Road
Campbell, CA 95008
tel: (408) 371-5100
fax: (408) 371-5638

William Wang 442

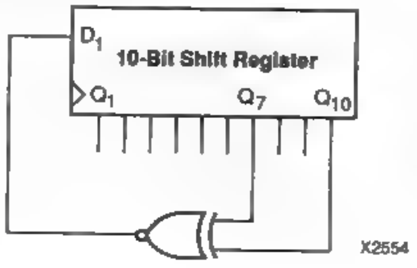
Linear Feedback Shift Register Counters

Conventional binary counters use complex or wide fan-in logic to generate high-end carry signals. A much simpler structure sacrifices the binary count sequence, but achieves very high speed with very simple logic, easily packing two bits into every CLB. Linear Feedback Register (LFSR) counters are also known as pseudo-random sequence generators.

An n-bit LFSR counter can have a maximum sequence length of 2^n-1. It goes through all possible code permutations except one, which is a lock-up state. A maximum length n-bit LFSR counter consists of an n-bit shift register with an XNOR in the feedback path from the last output Qn to the first input D1. (The XNOR makes the lock-up state the all-ones state, an XOR would make it the all-zeros state. For normal Xilinx applications, all-ones is preferred, since the flip-flops wake up in the all-zeros state.)

The table below describes the outputs that must drive the inputs of the XNOR. A mutli-input XNOR is also known as an even-parity circuit.

Note that the connections described in this table are not necessarily unique. Due to the symmetry of the shift register operation and the XNOR function, other connections may also result in maximum length sequences.



n	XNOR Feedback from Outputs
3	3,2
4	4,3
5	5,3
6	6,5
7	7,6
8	8,6,5,4
9	9,5
10	10,7
11	11,9
12	12,6,4,1
13	13,4,3,1
14	14,5,3,1
15	15,14
16	16,15,13,4
17	17,14
18	18,11
19	19,6,2,1
20	20,17
21	21,19
22	22,21
23	23,18
24	24,23,22,17
25	25,22
26	26,6,2,1
27	27,5,2,1
28	28,25
29	29,27
30	30,6,4,1
31	31,28
32	32,22,2,1
33	33,20
34	34,27,2,1
35	35,33
36	36,25
37	37,5,4,3,2,1
38	38,6,5,1
39	39,35
40	40,5,4,3

Examples

- A 10-bit shift register counts modulo 1023, if the input D1 is driven by the XNOR of Q10 and the bit three positions to the left (Q7), i.e. a one is shifted into D1 when Q10 and Q7 have even parity, which means they are identical.
- An 8-bit shift register counts modulo 255 if the input D1 is driven by the XNOR of Q8, Q6, Q5, Q4, i.e., a one is shifted into D1 if these four outputs have even parity, (four zeros, or two ones, or four ones).

XC4010 Pinout Update for PQ208

XC4010 Pinouts

Pin Description	PG191	PQ208
VCC	J4	183
I/O (A8)	J3	184
I/O (A9)	J2	185
I/O	J1	186
I/O	H1	187
I/O	H2	188
I/O	H3	189
I/O (A10)	G1	190
I/O (A11)	G2	191
I/O	F1	192
I/O	E1	193
GND	G3	194
I/O	F2	195
I/O	D1	196
I/O	C1	197
I/O	E2	198
I/O (A12)	F3	199
I/O (A13)	D2	200
I/O	B1	201
I/O	E3	202
I/O (A14)	C2	203
SGCK1 (A15, I/O)	B2	204
VCC	D3	205
GND	D4	2
PGCK1 (A16, I/O)	C3	4
I/O (A17)	C4	5
I/O	B3	6
I/O	C5	7
I/O (TDI)	A2	8
I/O (TCK)	B4	9
I/O	C6	10
I/O	A3	11
I/O	B5	12
I/O	B6	13
GND	C7	14
I/O	A4	15
I/O	A5	16
I/O (TMS)	B7	17
I/O	A6	18
I/O	C8	19
I/O	A7	20
I/O	B8	21
I/O	A8	22
I/O	B9	23
I/O	C9	24
GND	D9	25
VCC	D10	26
I/O	C10	27
I/O	B10	28
I/O	A9	29
I/O	A10	30
I/O	A11	31
I/O	C11	32
I/O	B11	33
I/O	A12	34
I/O	B12	35
I/O	A13	36
GND	C12	37
I/O	B13	38
I/O	A14	39
I/O	A15	40
I/O	C13	41
I/O	B14	42
I/O	A16	43
I/O	B15	44
I/O	C14	45
I/O	A17	46
SGCK2 (I/O)	B16	47
M1	C15	48
GND	D15	49
M0	A18	50
VCC	D16	55
M2	C16	56
PGCK2 (I/O)	B17	57
I/O (HDC)	E16	58
I/O	C17	59
I/O	D17	60
I/O	B18	61
I/O (LDC)	E17	62
I/O	F16	63
I/O	C18	64
I/O	D18	65
I/O	F17	66
GND	G16	67
I/O	E18	68
I/O	F18	69
I/O	G17	70
I/O	G18	71
I/O	H16	72
I/O	H17	73
I/O	H18	74
I/O	J18	75
I/O	J17	76
I/O (ERR, INIT)	J16	77
VCC	J15	78
GND	K15	79
I/O	K16	80
I/O	K17	81
I/O	K18	82
I/O	L18	83
I/O	L17	84
I/O	L16	85
I/O	M18	86
I/O	M17	87
I/O	N18	88
I/O ·	P18	89
GND	M16	90
I/O	N17	91
I/O	R18	92
I/O	T18	93
I/O	P17	94
I/O	N16	95
I/O	T17	96
I/O	R17	97
I/O	P16	98
I/O	U18	99
SGCK3 (I/O)	T16	100
GND	R16	101
DONE	U17	103
VCC	R15	106
PROG	V18	108
I/O (D7)	T15	109
PGCK3 (I/O)	U16	110
I/O	T14	111
I/O	U15	112
I/O (D6)	V17	113
I/O	V16	114
I/O	T13	115
I/O	U14	116
I/O	V15	117
I/O	V14	118
GND	T12	119
I/O	U13	120
I/O	V13	121
I/O (D5)	U12	122
I/O (CS0)	V12	123
I/O	T11	124
I/O	U11	125
I/O	V11	126
I/O	V10	127
I/O (D4)	U10	128
I/O	T10	129
VCC	R10	130
GND	R9	131
I/O (D3)	T9	132
I/O (RS)	U9	133
I/O	V9	134
I/O	V8	135
I/O	U8	136
I/O	T8	137
I/O (D2)	V7	138
I/O	U7	139
I/O	V6	140
I/O	U6	141
GND	T7	142
I/O	V5	143
I/O	V4	144
I/O	U5	145
I/O	T6	146
I/O (D1)	V3	147
RCLK-BUSY/RDY	V2	148
I/O	U4	149
I/O	T5	150
I/O (D0, DIN)	U3	151
SGCK4 (DOUT, I/O)	T4	152
CCLK	V1	153
VCC	R4	154
TD0	U2	159
GND	R3	160
I/O (A0, WS)	T3	161
PGCK4 (A1, I/O)	U1	162
I/O	P3	163
I/O	R2	164
I/O (CS1, A2)	T2	165
I/O (A3)	N3	166
I/O	P2	167
I/O	T1	168
I/O	R1	169
I/O	N2	170
GND	M3	171
I/O	P1	172
I/O	N1	173
I/O (A4)	M2	174
I/O (A5)	M1	175
I/O	L3	176
I/O	L2	177
I/O	L1	178
I/O	K1	179
I/O (A6)	K2	180
I/O (A7)	K3	181
GND	K4	182

PC68 Pinout Discrepancy

Page 2-33 of our 1992 Data Book lists pin-outs for the 68 and 84 pin packages for XC3020, XC3030, and XC3042. Some designers anticipate a migration of their design from the PC84 to the smaller PC68 package, and they carefully use only those PC84 I/O pins that are also available in the PC68. Unfortunately, this PC84-to PC68 relationship differs between XC3020 and XC3030. (It differs on pins 12, 13, 14, 15, 16, 21, 22, 31, 32, 33, 40, and 41.) Our 1991Data Book describes it correctly only for the XC3020 while our 1992 Data Book describes it correctly only for the XC3030.

The table below gives the composite description. We apologize for any confusion caused by this documentation error.

XC3000 Family 68-Pin PLCC, 84-Pin PLCC and PGA Pinouts

68 PLCC		XC3020 XC3030, XC3042	84 PLCC	84 PGA
XC03030	XC3020			
10	10	PWRDN	12	B2
11	11	TCLKIN-I/O	13	C2
12	—	I/O*	14	B1
13	12	I/O	15	C1
14	13	I/O	16	D2
—	—	I/O	17	D1
15	14	I/O	18	E3
16	15	I/O	19	E2
—	16	I/O	20	E1
17	17	I/O	21	F2
18	18	VCC	22	F3
19	19	I/O	23	G3
—	—	I/O	24	G1
20	20	I/O	25	G2
—	21	I/O	26	F1
21	22	I/O	27	H1
22	—	I/O	28	H2
23	23	I/O	29	J1
24	24	I/O	30	K1
25	25	M1-RDATA	31	J2
26	26	M0-RTRIG	32	L1
27	27	M2-I/O	33	K2
28	28	HDC-I/O	34	K3
29	29	I/O	35	L2
30	30	LDC-I/O	36	L3
—	31	I/O	37	K4
—		I/O*	38	L4
31	32	I/O	39	J5
32	33	I/O	40	K5
33	—	I/O*	41	L5
34	34	INIT-I/O	42	K6
35	35	GND	43	J6
36	36	I/O	44	J7
37	37	I/O	45	L7
38	38	I/O	46	K7
39	39	I/O	47	L6
—	40	I/O	48	L8
—	41	I/O	49	K8
40		I/O*	50	L9
41		I/O*	51	L10
42	42	I/O	52	K9
43	43	XTL2(IN)-I/O	53	L11

68 PLCC XC3030 XC3020	XC3020 XC3030, XC3042	84 PLCC	84 PGA
44	RESET	54	K10
45	DONE-PG	55	J10
46	D7-I/O	56	K11
47	XTL1(OUT)-BCLKIN-I/O	57	J11
48	D6-I/O	58	H10
—	I/O	59	H11
49	D5-I/O	60	F10
50	CS0-I/O	61	G10
51	D4-I/O	62	G11
—	I/O	63	G9
52	VCC	64	F9
53	D3-I/O	65	F11
54	CS1-I/O	66	E11
55	D2-I/O	67	E10
—	I/O	68	E9
—	I/O*	69	D11
56	D1-I/O	70	D10
57	RDY/BUSY-RCLK-I/O	71	C11
58	D0-DIN-I/O	72	B11
59	DOUT-I/O	73	C10
60	CCLK	74	A11
61	A0-WS-I/O	75	B10
62	A1-CS2-I/O	76	B9
63	A2-I/O	77	A10
64	A3-I/O	78	A9
—	I/O*	79	B8
—	I/O*	80	A8
65	A15-I/O	81	B6
66	A4-I/O	82	B7
67	A14-I/O	83	A7
68	A5-I/O	84	C7
1	GND	1	C6
2	A13-I/O	2	A6
3	A6-I/O	3	A5
4	A12-I/O	4	B5
5	A7-I/O	5	C5
—	I/O*	6	A4
—	I/O*	7	B4
6	A11-I/O	8	A3
7	A8-I/O	9	A2
8	A10-I/O	10	B3
9	A9-I/O	11	A1

Unprogrammed IOBs have a default pull-up. This prevents an undefined pad level for unbonded or unused IOBs. Programmed outputs are default slew-rate limited.

This table describes the pinouts of three different chips in three different packages. The second column lists 84 of the 118 pads on the XC3042 (and 84 of the 98 pads on the XC3030) that are connected to the 84 package pins. Ten pads, indicated by an asterisk, do not exist on the XC3020, which has 74 pads; therefore the corresponding pins on the 84-pin packages have no connections to an XC3020. Six pads on the XC3020 and 16 pads on the XC3030, indicated by a dash (—) in the 68 PLCC column, have no connection to the 68 PLCC, but are connected to the 84-pin packages.

Programmable Gate Array Training Courses

The focus of our training department is to provide high-quality, comprehensive training for our customers on how to use our products, both software and ICs. We offer a variety of classes on different platforms and in various locations, thus providing a way for our customers to get up-to-speed and become productive as quickly and efficiently as possible.

We are offering two different classes on our XC3000 family products: A 2-day class gives a very good overview for beginning users, and a 4-day class that provides a more in-depth look at the architecture, development system features, and recommended design methodology. Both of these classes are for ALL users of XC2000 and XC3000 family products. The classes start at the introductory level, but quickly move into more detail, so that experienced users also find these classes beneficial.

We also offer a 2-day XC4000 family class that is intended for designers who are already familiar with our XC2000 or XC3000 family devices. It covers the XC4000 architecture, development system features, and recommended design methodology.

For new XC4000 customers that are not familiar with Xilinx FPGAs, we recommend a 4-day XC3000/XC4000 family combination course. This introductory class covers all of our products, and, shows how they fit together. The background information learned is invaluable, since parts of some applications may well fit into an XC3000 device, possibly providing a more cost-effective solution.

All of our classes include hands-on lab exercises giving you the opportunity to gain the experience you need to be productive immediately after you return to your office. Most of the classes use PCs, but some of the training centers also have SPARCstations. Our focus is on the process of designing, not on the specific platform or schematic entry tools.

We offer these classes in our factory in San Jose, California, and also in Regional Training Centers worldwide. Please consult the schedule on the next page for classes in your area. We can provide any of these classes, or custom-tailored classes, at you own facility. If you have any questions, please contact your local Xilinx sales office.

The standard price of the classes is $1000 per student for the 4-day classes, and $750 per student for the 2-day classes. These are US prices, and vary in the international locations.

RR

Course Programs

XC3000

Day 1 XACT Design Manager
XMAKE Automatic Translation - *Lab*
Basic Architecture
Estimating Size
Design Entry - *Lab*

Day 2 Design Implementation
XNFMAP Partitioning - *Lab*
APR Placement and Routing - *Lab*
MAKEBITS Bitstream Generator
MAKEPROM PROM Formatter

Day 3 Configuration
Design Verification
Simulation - *Lab*
Downloading

Day 4 EDITLCA Graphical Editor - *Lab*
Architecture Details - *Lab*

XC4000

Day 1 XACT Design Manager
XMAKE Automatic Translation - *Lab*
Basic Architecture
Estimating Size
Design Entry - *Lab*
MEMGEN RAM/ROM Compiler - *Lab*

Day 2 Design Implementation
PPR Partitioning, Placement, & Routing
Configuration - *Lab*
MAKEBITS Bitstream Generator - *Lab*
MAKEPROM PROM Formatter
Design Verification
Downloading - *Lab*
Readback
EDITLCA Graphical Editor - *Lab*

XC3000 & XC4000

Day 1 XACT Design Manager
XMAKE Automatic Translation - *Lab*
XC3000 Basic Architecture
XC3000 Estimating Size
XC3000 Design Entry - *Lab*

Day 2 XC3000 Design Implementation - *Lab*
MAKEBITS Bitstream Generator
MAKEPROM PROM Formatter
Downloading
XC3000 Configuration
Design Verification

Day 3 EDITLCA Graphical Editor - *Lab*
XC4000 Basic Architecture
XC4000 Estimating Size
XC4000 Design Entry - *Lab*

Day 4 XC4000 MEMGEN
RAM/ROM Compiler - *Lab*
XC4000 Design Implementation
XC4000 Configuration - *Lab*
XC4000 Downloading - *Lab*

Training Schedule

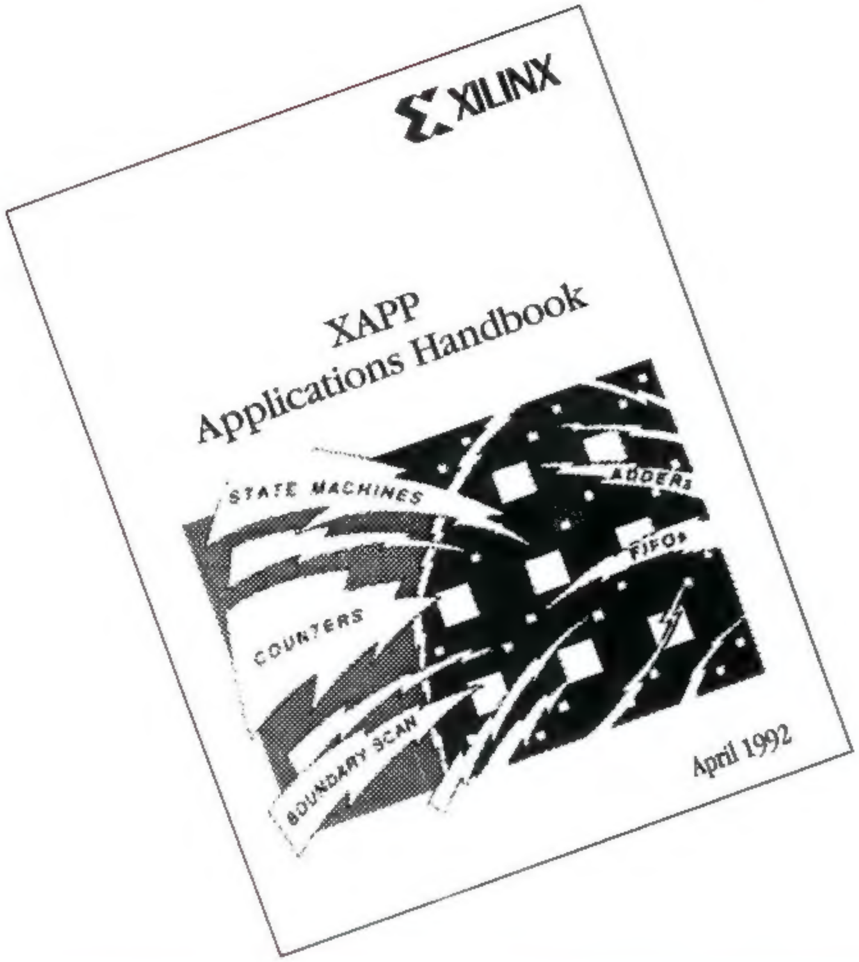
	July		August		September		October		November		December	
	Date	Class	Date	Class	Date	Class	Date	Class	Date	Class	Date	Class
San Jose, CA Xilinx Headquarters Training Coordinator: Jennifer Donnell, (408) 879-5090	6-9 20-21 27-30	XC3000 XC4000 Combined	10-13 17-18 24-27	XC3000 XC4000 Combined	14-17 21-22 28-10/1	XC3000 XC4000 Combined	5-8 9-10 16-19	XC3000 XC4000 Combined	2-5 9-10 16-19 30-12/3	XC3000 XC4000 Combined XC3000	7-8 14-17	XC4000 Combined
Southern California Hamilton/Avnet	15-16 San Diego	XC3000	12-13 Costa Mesa	XC3000	9-10 Woodland Hills	XC3000	21-23 San Diego	Combined	11-13 Costa Mesa	Combined	9-11 Woodland Hills	Combined
Phoenix, AZ Hamilton/Avnet							7-9	Combined				
Westboro, MA Massachusetts Microelectronics Center			3-6	Combined	21-22	XC3000	19-22	Combined	9-12	Combined	7-8	XC3000
Longmont, CO SIS Microelectronics			20-21	XC3000					19-20	XC3000		
Rochester, NY Rochester Institute of Technology	14-17 Sun Workstation	Combined	18-21 Sun Workstation	Combined					17-20 Sun Workstation	Combined		
Boca Raton, FL Florida Atlantic University			Call	Combined							Call	Combined
Chicago, IL Illinois Institute of Technology			18-21	Combined			Call	Combined				
Arlington, TX University of Texas			4-7 Sun Workstation	Combined					15-18 Sun Workstation	Combined		
Germany Metronik: (49) 89 611 080 D = Darmstadt, F = Freiburg, M = München	8-10 M 20-24 D 27-29 D	XC4000 XC3000 XC4000	19-21 F 24-28 M	XC4000 XC3000	9-11 M 14-18 M 28-30 M	XC4000 XC3000 XC3000	5-7 M 14-16 M 26-30 M	XC4000 XC4000 XC3000	4-6 M 9-11 M 23-27 M 25-27 D	XC4000 XC3000 XC3000 XC4000	7-11 F 14-16 F 16-18 M	XC3000 XC4000 XC3000
Paris, France ESIEE: (33) 1 45 92 66 01					21-24	Combined			23-26	Combined		
Breda, Netherlands Rodelco BV Electronics: (31) 76 784 911					22-24	XC3000			24-26	XC3000		
Tokyo/Osaka, Japan Inoware 21: (81) 35695 1521	Call Call Call	XC3000 XC4000 Combined	Call Call Call	XC3000 XC4000 Combined	Call Call Call	XC3000 XC4000 Combined	Call Call Call	XC3000 XC4000 Combined	Call Call Call	XC3000 XC4000 Combined	Call Call Call	XC3000 XC4000 Combined
Hong Kong Excel Associates: (852) 418-0909			Call	XC3000			Call	XC3000			Call	XC3000

For information on scheduled classes, call: Southern California/Phoenix, AZ – Nikki Gordon at Hamilton/Avnet, (714) 641-4198
Westboro, MA – Paulette Renaud at M2C, (508) 870-0312
Longmont, CO – Sheree Carter at Luscombe Engineering, (303) 772-3342

For other U.S. locations, or to schedule an in-house class, call Jennifer Donnell, the Xilinx Training Coordinator, at (408) 879-5090.

For non-U.S. locations or in-house classes, call the numbers shown in the table.

Applications Handbook



As a part of an on-going commitment to customer support, Xilinx has just published the first XAPP Applications Handbook. The first edition contains 16 application notes that address a wide range of topics; more application notes are already in progress.

The handbook contains in-depth explanations of LCA features and design examples, showing how best to utilize the LCA device. Most of the design examples have been fully implemented, and View*logic*[*] schematic files are available.

These files illustrate the complete implementation process, and provide macros that may be used in other designs. Copies of these design files can be obtained through the Xilinx Technical Bulletin Board, or by calling the Xilinx Applications Hotline for a disk.

Contact Xilinx for your copy of the XAPP Applications Handbook.

[*] View*logic* is a trademark of Viewlogic Systems, Inc.

XILINX

2100 Logic Drive
San Jose, CA 95124-3450

U.S. POSTAGE
PAID
FIRST CLASS
PERMIT NO. 2196
SAN JOSE, CA